Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Artificial Intelligence 145 (2003) 1–32

Artificial Intelligence

www.elsevier.com/locate/artint

Multiobjective heuristic state-space planning

Ioannis Refanidis [a,*], Ioannis Vlahavas [b]

[a] *University of Macedonia, Department of Applied Informatics 54006, Thessaloniki, Greece*
[b] *Aristotle University, Department of Informatics 54124, Thessaloniki, Greece*

Received 2 April 2001; received in revised form 12 December 2001

Abstract

Modern domain-independent heuristic planners evaluate their plans on the single basis of their length. However, in real-world problems, there are other criteria that also play an important role, e.g., resource consumption, profit, safety, etc. This paper enhances the GRT planner, an efficient domain-independent heuristic state-space planner, with the ability to consider multiple criteria. The GRT heuristic is based on the estimation of the distances between each fact of a problem and the goals. The new planner, called MO-GRT, uses a weighted A* strategy and a multiobjective heuristic function, computed over a weighted hierarchy of user-defined criteria. Its computation is based on sets of non-dominated cost-vectors assigned to the problem facts, which estimate the total cost of achieving the facts from the goals, using alternative paths. Experiments show that a change in the criteria weights or scales affects both the quality of the resulting plan and the planning time. The proposed approach can easily be adapted to other modern heuristic state-space planners.

Keywords: Planning; Heuristic search; Multiple criteria; Multiobjective search

1. Introduction

In modern planning systems, especially in the domain-independent ones, it is common practice to evaluate plans on the basis of a single criterion, i.e., plan length. In order to support this statement, it suffices to mention the domains used in the last two planning competitions, both in the domain-independent and in the domain-dependent tracks [1,21]. In any case, the objective was to obtain short plans. However, most real-world problems

* Corresponding author.
E-mail addresses: yrefanid@uom.gr (I. Refanidis), vlahavas@csd.auth.gr (I. Vlahavas).

0004-3702/02/$ – see front matter
doi:10.1016/S0004-3702(02)00371-5

require the consideration of other, often contradictory, criteria as well, e.g., plan duration, fuel consumption, profit, safety, etc.

This paper presents MO-GRT, a domain independent heuristic state-space planner, which is able to take multiple criteria into account. The word criterion refers to any type of measurable quantity, which is of interest in the solution plan. These criteria are provided by the user, along with the definition of the problem. The kind of problems that MO-GRT handles successfully are characterized by linear aggregation of the criteria values and by the ability to set bounds on them. Moreover, MO-GRT can handle both monotonic and non-monotonic criteria.

MO-GRT is based on the single-objective planner GRT [28,31], an efficient heuristic state-space planner, working under the known STRIPS [9] framework. The GRT heuristic is constructed in a domain-independent way. The basis of this construction is the estimation of the distance between each individual fact of a problem and the goals. The obtained estimates are further used to estimate the distance between each state and the goals, thus leading the search process to a forward direction. Experiments have shown that GRT achieves good performance in many domains [32].

In order to take multiple criteria into account, the new heuristic assigns a set of cost-vectors to each fact. A cost-vector is an estimate of the total cost of achieving the fact by applying actions to the goal state, whereas its elements correspond to the various criteria. Different vectors for the same fact correspond to alternative ways of achieving the fact. The states are evaluated using both the known accumulated value of the past plan, and the estimated value of the remaining plan based on the cost-vectors of the state's facts. The search-space is traversed using a weighted A* strategy, which enables the planner to exchange planning time and plan quality.

The multiobjective heuristic search paradigm has been introduced by Stewart and White [33], who extended the typical A* algorithm in a vector-valued state space. The foundations of the multiobjective search on AND/OR graphs and game trees have been presented in [7] and [8], respectively. Applications of the multiobjective framework in planning are also found in [10,24,36]. However, these works assumed a given domain-dependent heuristic function. Besides, in most cases, an attempt was made to find all the non-dominated solutions using exhaustive enumeration and evaluation of all states of the search space. Our approach is different in that it deals with the construction of a vector-valued heuristic function for STRIPS-type planning problems in a domain-independent way. Moreover, it supports the definition of preferences among the several criteria, allowing tradeoffs among them. Finally, the aim is not to find all non-dominated solutions; it is to find the best compromise between the solution quality and the available planning time.

This paper is structured as follows: Section 2 introduces the main planning concepts and Section 3 defines the multiobjective planning problem and provides useful properties of the evaluation functions. Section 4 briefly presents the single-objective GRT planner algorithm and structures. The MO-GRT planner is thoroughly examined in Section 5, while Section 6 presents a number of performance measurements in a logistics-like domain, where multiple criteria have been defined. Section 7 outlines the adaptation of the multiobjective framework to other heuristic state-space planners. Section 8 presents related work and, finally, Section 9 summarizes the paper and identifies future challenges.

2. Preliminaries

In STRIPS [9], each action a is represented by three sets of facts: the precondition list $Pre(a)$, the add list $Add(a)$ and the delete list $Del(a)$, where $Del(a) \subseteq Pre(a)$. A state S is defined as a finite set of facts. An action a is applicable to a state S if $Pre(a) \subseteq S$. The state resulting from the application of a to S is defined as:

$$S' = res(S, a) = S \cup Add(a) - Del(a) \tag{1}$$

Inductively, we can define the state resulting from the application of a sequence of M actions $(a_1, a_2, \ldots, a_M)$ to a state S as:

$$S' = res\big(S, (a_1, a_2, \ldots, a_M)\big) = res\big(res\big(S, (a_1, a_2, \ldots, a_{M-1})\big), a_M\big) \tag{2}$$

with the requirement that each action a_i is applicable to the state $res(S, (a_1, a_2, \ldots, a_{i-1}))$, for each $i = 1, 2, \ldots, M-1$.

A planning problem P can be defined as a triplet $P = (A, Initial, Goals)$, where A is a set of ground actions, *Initial* is a state and *Goals* is a set of facts. The task is to find a sequence of actions $a_1, a_2, \ldots, a_M$ that can be applied to *Initial*, so that the state resulting from their application will be a superset of *Goals*.

$$Goals \subseteq res\big(Initial, (a_1, a_2, \ldots, a_M)\big) \tag{3}$$

The sequences of actions are called *plans*. A plan that can be applied to the initial state is called *valid plan*. A valid plan that achieves *Goals* is called *solution* of the planning problem. A planning problem may have several solutions or none.

In single objective planning, a cost $c(a)$ is assigned to each action a. This cost may be interpreted in several ways, namely the duration of the action, its application cost or profit, etc. Similarly, an overall cost can be assigned to a plan, which is defined as the sum of costs of its individual actions, i.e., $c((a_1, a_2, \ldots, a_M)) = \sum_{i=1}^{M} c(a_i)$. In many real-world planning problems we are not interested in finding any solution, but in finding the optimal one, i.e., the one that minimizes (or maximizes, depending on the interpretation of the notion of cost) the overall cost. In case that finding optimal solutions is computationally too expensive, it may be acceptable to find a near optimal one. However, near-optimality is a subjective notion and cannot always be well defined, since, in most cases, it is not known in advance what is the cost of an optimal solution.

State-space planners proceed by repeatedly applying actions to the initial state and to the resulting states, thus keeping a frontier set of states. There are several search algorithms determining the order in which the states of the frontier set are explored. To make search more efficient, it is usually possible to use heuristic functions, which estimate the cost between each state of the frontier set and the goals, thus selecting to explore first the most promising ones. Heuristic functions that do not overestimate the cost of reaching the goals from any state are described as admissible. The combination of an admissible heuristic function with the A* search algorithm [12] ensures that the first obtained solution will be an optimal one.

3. The multiobjective planning problem

In a multiobjective planning problem, a vector of costs $\mathbf{v}(a)$[1] is assigned to each action a. In this case, the overall cost of a plan $(a_1, a_2, \ldots, a_M)$ is also a vector $\mathbf{v}((a_1, a_2, \ldots, a_M)) = \sum_{i=1}^{M} \mathbf{v}(a_i)$. Comparing plans in the multiobjective framework is not simple, since for any two plans it may be possible that each one of them is better than the other one in terms of the different dimensions of their cost-vectors, e.g., the one may have shorter execution time, whereas the other may have lower execution cost.

Therefore, in order to compare plans, an evaluation function E has to be defined over the space of the cost-vectors. In this paper we only consider arithmetic functions; however, any other function capable of producing acyclic orderings between any arbitrary set of cost-vectors could be transformed into an arithmetic one.

Definition 1 (*Plan comparison*). A solution plan P_1 is considered better than a solution plan P_2 for a given evaluation function E, iff $E(\mathbf{v}(P_1)) < E(\mathbf{v}(P_2))$.

Note that we consider the lowest values of the evaluation function best. However, this commitment does not reduce the generality of the proposed method, since any other evaluation function E_1 considering the highest values best could be transformed to an equivalent function E_1' that satisfies Definition 1, just by multiplying with -1, i.e., $E_1' = -1 \cdot E_1$. In the following paragraphs some desired properties that may have an evaluation function are presented.

Definition 2 (*Additive property*). An evaluation function E satisfies the additive property, iff for any two cost-vectors $\mathbf{v}$ and $\mathbf{u}$ the cost of their sum is equal to the sum of their costs, i.e.:

$$E(\mathbf{v} + \mathbf{u}) = E(\mathbf{v}) + E(\mathbf{u}), \quad \forall \mathbf{v}, \mathbf{u} \tag{4}$$

Proposition 3. *An evaluation function satisfies the additive property iff it has a linear form and the valid values of the individual dimensions of the cost-vectors are unbounded.*

For example, the function:

$$E(\mathbf{v}) = a_1 v_1 + a_2 v_2 + \cdots + a_k v_k \tag{5}$$

where v_i are the dimensions of $\mathbf{v}$ (i ranges from 1 to k) and a_i are constant values, satisfies the additive property in case the values of v_i are unbounded. On the other hand, functions with non-linear forms do not satisfy the additive property. Even linear functions, which cannot be defined for any arbitrary cost-vector, do not satisfy the additive property, since the vector $\mathbf{v} + \mathbf{u}$ in Eq. (4) may not be definable. This situation usually arises in case some of the dimensions of the cost-vectors are bounded, e.g., there is a strict upper or lower bound.

[1] Throughout the text bold typeface is used to denote vectors.

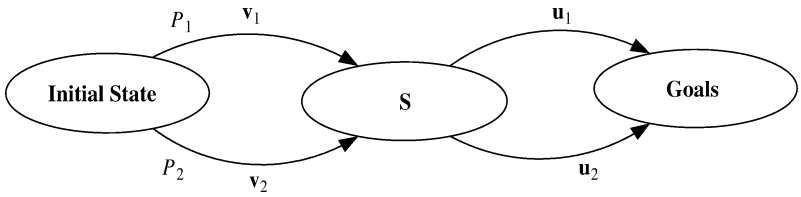

Fig. 1. Evaluating a state in the presence of many criteria.

Corollary 4. *For two sets of cost-vectors* **V** *and* **U** *and for an evaluation function* E *that satisfies the additive property, the minimum cost among the sum of any pair of cost-vectors* $\mathbf{v} \in \mathbf{V}$ *and* $\mathbf{u} \in \mathbf{U}$ *is equal to the sum of the minimum costs of* **V***'s and* **U***'s cost-vectors, i.e.*:

$$\min_{\mathbf{v}\in\mathbf{V},\mathbf{u}\in\mathbf{U}}\big(E(\mathbf{v})+E(\mathbf{u})\big)=\min_{\mathbf{v}\in\mathbf{V}}\big(E(\mathbf{v})\big)+\min_{\mathbf{u}\in\mathbf{U}}\big(E(\mathbf{u})\big) \tag{6}$$

In case of an evaluation function satisfying the additive property, it is possible to adopt a single-objective approach to solve the planning problem, just by assigning the value $E(\mathbf{v}(a))$ to each action a. However, this approach cannot be adopted in case the evaluation function does not satisfy the additive property, which is usual. This is illustrated in Fig. 1.

Suppose there is a state S of the state space, which can be reached by the initial state following two alternative paths, P_1 and P_2, with costs $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively, so that $E(\mathbf{v}_1) < E(\mathbf{v}_2)$. In case of an additive evaluation function, path P_2 could safely be omitted, since, for any path leading from S to the goals with cost $\mathbf{u}$, $E(\mathbf{v}_1 + \mathbf{u}) < E(\mathbf{v}_2 + \mathbf{u})$ (Corollary 4). Similarly, suppose there are two heuristic estimations $\mathbf{u}_1$ and $\mathbf{u}_2$ for the cost of reaching the goals from S, with $E(\mathbf{u}_1) < E(\mathbf{u}_2)$. In this case, the estimation $\mathbf{u}_2$ could be omitted, since, for any path P leading from the initial state to S with cost $\mathbf{v}$, $E(\mathbf{v}+\mathbf{u}_1) < E(\mathbf{v}+\mathbf{u}_2)$ (Corollary 4). The situation described above is identical with what is always the case in single-objective state-space planning, e.g., for each state, we need a single cost (and the corresponding path) to reach the state from the initial state and a single cost estimation to reach the goals from that state.

However, in case the evaluation function is not additive, we cannot omit neither the alternative paths between the initial state and any intermediate state nor the heuristic estimations of the cost of reaching the goals from any intermediate state. Thus, a large number of cost-vectors may have to be stored for each state, thus increasing the difficulty to solve the planning problem.

Another desirable property of an evaluation function is the monotonicity property. In order to define this property, we firstly introduce the domination relation between two cost-vectors **v** and **u**, denoted with $\prec$.

Definition 5 (*Domination*). A cost-vector **v** is said to dominate another vector **u** and this is denoted by $\mathbf{v} \prec \mathbf{u}$, if for each i, $1 \leqslant i \leqslant N$, v_i is better than u_i and $\mathbf{v} \neq \mathbf{u}$ (N stands for the vector dimensions).

The characterization "is better than" in Definition 5 may be interpreted either as "is lower than" or as "is larger than", depending on the nature of each dimension of the cost-

vectors. For example, for a dimension denoting duration, lower values are considered best, whereas for a dimension denoting profit, higher values are considered best.

Definition 6 (*Non-dominated vector*). A vector $\mathbf{v}$ is described as non-dominated, if there is no other vector dominating $\mathbf{v}$.

Having defined the concept of domination, we can introduce the monotonicity property for functions over vectors, extending the known monotonicity property of functions over real values.

Definition 7 (*Monotonicity property*). An evaluation function E is described as monotonic, iff $\forall \mathbf{v}, \mathbf{u}$: $\mathbf{v} \prec \mathbf{u} \Rightarrow E(\mathbf{v}) < E(\mathbf{u})$.

It is not difficult to prove that an evaluation function over a set of criteria satisfies the monotonicity property, iff its first derivatives on these criteria are either positive or negative functions, depending on whether the lowest or the highest values of the various criteria are considered best, respectively.

The monotonicity property is satisfied by the evaluation functions of most real-world planning problems and helps to reduce the complexity of the planning process. This is achieved by leaving out all cost-vectors dominated by other vectors, thus keeping only the non-dominated ones. For example, if in Fig. 1 $\mathbf{v}_1 \prec \mathbf{v}_2$ is true and the evaluation function is monotonic, we can safely omit $\mathbf{v}_2$. Similarly, if $\mathbf{u}_1 \prec \mathbf{u}_2$, we can also omit $\mathbf{u}_2$. This pruning is based on the fact that if $\mathbf{v}_1 \prec \mathbf{v}_2$, then for any other cost-vector $\mathbf{u}$ it is true that $\mathbf{v}_1 + \mathbf{u} \prec \mathbf{v}_2 + \mathbf{u}$ and, consequently, $E(\mathbf{v}_1 + \mathbf{u}) < E(\mathbf{v}_2 + \mathbf{u})$. Note that in case $\mathbf{v}_1 + \mathbf{u}$ cannot be defined due to the hard bounds of some dimensions, $\mathbf{v}_2 + \mathbf{u}$ would not be defined either, since the values of $\mathbf{v}_2$ are worse than $\mathbf{v}_1$ and the bounds are usually set in the worst values (there is no reason to set bounds in the best values). Keeping only the non-dominated cost-vectors results in a reduction of the complexity of the planning process; however, their number remains large. It is not difficult to show that all linear-form functions satisfy the monotonicity property.

Note finally that in case the heuristic function underestimates the cost of reaching the goals from any intermediate state and for various criteria, we can use its estimates to prune possible expansions of the states in the frontier set, based on the bounds of these criteria. Therefore, admissible heuristic functions can possibly further reduce the complexity of the planning process.

4. The single-objective GRT planner

The GRT planner is a domain-independent heuristic state-space planner [28,31], where the term "domain-independent" refers to the way the heuristic function is constructed, i.e., a single algorithm is used for all domains. Its heuristic function estimates the distance, in terms of the number of actions, between any state and the goals, thus trying to minimize the plan length. However, the heuristic function is not admissible and GRT does not use an A* search strategy; instead, it either adopts the best-first search or the hill-climbing one.

Thus, GRT, like all other effective heuristic planners, does not guarantee to find optimal plans.

The planning process consists of two phases: the heuristic construction phase and the search phase. In the first phase, GRT estimates the distance between each fact of the planning problem and the goals. This is performed by repeatedly applying actions to the goals until all problem facts have been achieved, and by using the number of actions needed to achieve each fact as its distance estimation. The data obtained during this phase is stored in a table, the Greedy Regression Table, which is indexed by the problem facts. Then, in the search phase, the planner proceeds forward from the initial state to the goals and uses the entries of this table to further estimate the distance between each state of the state-space and the goals.

A difficulty that may arise in the heuristic construction phase is that the actions of a problem cannot always be applied to the goals. GRT solves this problem by computing and using the inverted actions instead. Suppose there is an action $a \in A$ and two states s and s', so that a can be applied to s and $s' = res(s, a)$. The inverted action a' of a is an action, so that $s = res(s', a')$. The inverted action is defined by the original action using the following formulas:

$$\begin{aligned} Pre(a') &= Add(a) + Pre(a) - Del(a) \\ Del(a') &= Add(a) \\ Add(a') &= Del(a) \end{aligned} \tag{7}$$

Note that, in many problems, especially in the benchmark problems used in the recent planning competitions [1,21], the set of inverted actions is identical with the set of normal ones.

Another difficulty that may arise in the heuristic construction phase is that in some planning problems, the goals do not constitute a complete state description, so it is not even possible to apply the inverted actions to them. GRT solves this problem by detecting the candidate missing goal facts and enhancing the goals with some or all of them, in a fully automated way [29,31].

The distance $dist(p)$ between a fact p and the goals is estimated by the following recursive formula:

$$dist(p) = \min \begin{cases} 0, & \text{if } p \in Goals. \\ \text{AGGREGATE}(Pre(a')) + 1, & \\ \qquad \text{where } a' \text{ is an inverted} & \\ \qquad \text{action such that } p \in Add(a'). & \\ \infty, & \text{otherwise} \end{cases} \tag{8}$$

In formula (8) the prefix operator min operates on sets of numbers and returns the minimum of them. The recursion follows from function AGGREGATE, which uses the distances of its arguments in order to produce its result, as it will be shown later in this section. Formula (8) is repeatedly applied until all distances stabilize. The function AGGREGATE takes as input a set of already achieved facts and their individual distances and returns the cost of achieving them simultaneously. This function can be defined in several ways. For example, AGGREGATE could return either the sum or the maximum of the distances of its arguments. Interpreting AGGREGATE as a sum function produces very informative estimates, which however usually overestimate the actual distances; thus, they cannot

be used for pruning purposes. On the other hand, interpreting AGGREGATE as a max function results always in underestimates; however, these are not informative and not useful in the planning process [6].

In order to obtain more accurate and informative estimates, GRT introduces the notion of related facts. A fact q is considered related to another fact p, if the achievement of p leads to the achievement of q as well. The facts related to a specific fact p are called related facts of p and are denoted by $related(p)$. Intuitively, we can define the related facts of a set of facts P as the union of the related facts of P's facts, i.e., $related(P) = \bigcup_{p \in P} related(p)$. For an inverted action a' achieving a fact p, the related facts of p are defined by the following recursive formula:

$$related(p) = Pre(a') \cup related\big(Pre(a')\big) \cup Add(a') - Del(a') - \{p\} \tag{9}$$

which is initialized for the goal facts by setting $related(g) = \emptyset$, for each $g \in Goals$.

The related facts of a fact p depend on the specific path, i.e., the sequence of actions followed to achieve p. Since there are many paths to achieve a specific fact, there are many ways to define its related facts. For efficiency, GRT considers a single set of related facts corresponding to a path with minimum distance, for each fact. In case there are many alternative paths with the same minimum distance, GRT selects one of them arbitrarily.

Related facts play a critical role in the estimation of the distances between each fact and the goals. Thus, GRT considers the function AGGREGATE a combination of sum and max functions, which groups its argument facts in disjoint sets of related facts and sums the maximum distances of each group. The full definition of the function AGGREGATE, as implemented in GRT, is the following:

Function AGGREGATE.
Input: A set of facts $\{p_1, p_2, \ldots, p_N\}$, their distances $dist(p_i)$ and their lists of related facts $rel(p_i)$.
Output: An estimate of the cost of achieving the facts simultaneously.

1. Set $M_1 = \{p_1, p_2, \ldots, p_N\}$. Set $Cost = 0$.
2. While $(M_1 \neq \emptyset)$ do:
   (a) Let M_2 be the set of facts $p_i \in M_1$ that are not included in any list of related facts of another fact $p_j \in M_1$, without p_j being also included in their list of related facts. More formally:
   $M_2 = \{p_i\colon\ p_i \in M_1, \forall p_j \in M_1, p_i \in rel(p_j) \Rightarrow p_j \in rel(p_i)\}$
   (b) Let M_3 be the set of those facts of M_1 that are not included in M_2, but are included in at least one of the lists of related facts of the elements of M_2.
   $M_3 = \{p_i\colon\ p_i \in M_1 - M_2, \exists p_j \in M_2, p_i \in rel(p_j)\}$
   (c) Divide M_2 in disjoint groups of facts that are related to each other. For each group add the common cost of its facts to *Cost*.
   (d) Set $M_1 = M_1 - M_2 - M_3$.
3. Return *Cost*.

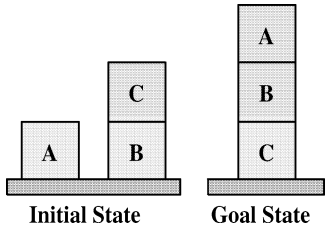

Fig. 2. A 3-blocks problem.

Table 1
Part of the Greedy Regression Table for the 3-blocks problem

#	Fact	Distance from goals	Related facts
1	(*on C table*)	0	–
2	(*on B C*)	0	–
3	(*on A B*)	0	–
4	(*clear A*)	0	–
5	(*on A table*)	1	(*clear B*) (*clear A*)
6	(*clear B*)	1	(*on A table*) (*clear A*)
7	(*on B table*)	2	(*clear C*) (*clear B*) (*on A table*) (*clear A*)
8	(*clear C*)	2	(*on B table*) (*clear B*) (*on A table*) (*clear A*)
9	(*on C B*)	3	(*clear C*) (*on B table*) (*on A table*) (*clear A*)
…	…	…	…

In [31] it was shown that the set M_2 (step 2(a) of function AGGREGATE) will never be empty and it can always be partitioned in disjoint groups of facts achieved by the same action (step 2(c)). The number of iterations the function AGGREGATE performs is bounded by the initial size of M_1; however, one or two iterations are usually performed.

The function AGGREGATE is also used in the search phase of the planning process, in order to estimate the cost of achieving the goals from any state of the state-space. In this case, the function is applied to the state facts and the resulting value is the estimate.

Example. The overall operation of GRT is illustrated with the *blocks-world* problem shown in Fig. 2. Part of the Greedy Regression Table for this problem is shown in Table 1.

Let us present first how Table 1 has been constructed. In this problem, the goal state is a complete state, so there is no problem in applying actions to it. Moreover, the set of inverted actions is identical to the set of normal actions, thus we will use the normal actions for simplicity.

Initially, zero distances and empty sets of related facts are assigned to the goal facts, i.e., (*on C table*), (*on B C*), (*on A B*) and (*clear A*)[2] (Table 1, rows 1 to 4). The first ground action that can be applied is *move-A-from-B-to-table*, which has an application cost equal to 1 (since its preconditions have zero distances). This cost is assigned to the facts that the action achieves, e.g., (*on A table*) and (*clear B*). Moreover, these facts are considered

[2] For the representation of facts and states a LISP-like notation is adopted throughout this paper.

related to each other, as well as to the precondition fact (*clear A*), which is not deleted by the action (Table 1, rows 5 and 6).

Suppose that next the action *move-B-from-C-to-table* is attempted, which has the facts (*clear B*) and (*on B C*) as preconditions. The facts are not related, so the application cost of the action is 2 and is assigned to the facts that the actions achieves, e.g., (*on B table*) and (*clear C*). These two facts are related to each other as well as to the precondition fact (*clear B*) and its related facts (*on A table*) and (*clear A*), since all these facts are not deleted by the action (Table 1, rows 7 and 8).

Finally, suppose that next the action *move-C-from-table-to-B* is attempted, which has the facts (*on C table*), (*clear C*) and (*clear B*) as preconditions. It is not difficult to understand that, if function AGGREGATE is called with these three facts as arguments, it returns the value 2, thus the action has an application cost of $2 + 1 = 3$. This cost is assigned to the fact (*on C B*), which is achieved by the action. Moreover, the non-deleted preconditions of the action, along with their related facts, constitute the set of related facts of (*on C B*), according to formula (9).

This process will continue until no more problem facts can be achieved in a lower cost. In this problem, the costs computed above will not change. Let us compute now the distance between the initial and the goal state. The initial state consists of the following facts:

$$\big((\textit{on A table})\ (\textit{clear A})\ (\textit{on B table})\ (\textit{on C B})\ (\textit{clear C})\big)$$

As it results from Table 1, all the initial state facts are related to (*on C B*). Thus, in the first iteration of the AGGREGATION loop, M_2 is set to ((*on C B*)) (step 2a) and M_3 is set to ((*on A table*) (*clear A*) (*on B table*) (*clear C*)) (step 2b). Thus, *Cost* becomes equal to the distance of (*on C B*), i.e., 3 (step 2c) and M_1 becomes empty. A second iteration is not performed and value 3, which is the actual distance between the initial and the goal state, is returned.

It is not difficult to see that, in case the notion of related facts had not been introduced, the costs assigned to the problem facts would have been higher and finally the cost of reaching the goals from the initial state would have been estimated to the value 8. By exploiting the related facts, GRT manages to obtain estimates which are closer to the actual values, which is very useful in case these estimates are to be used for heuristic pruning purposes or even just to guide an A* search strategy. However, even with the related facts, there are cases where the estimates produced by the GRT heuristic are overestimates.

5. The multiobjective GRT planner

This section presents the MO-GRT planner in detail. The section starts with the definition of the criteria hierarchy, next presents the construction of the heuristic function in the presence of multiple criteria, next illustrates how the states are evaluated using a weighted A*-like approach, next presents an enhanced form of the planning graphs where resource-vectors are assigned to the fact nodes and, finally, concludes with an example.

5.1. Evaluation criteria

Plan evaluation criteria can be classified on the basis of several features. The first one refers to the higher or lower values that are considered best. For some criteria, as e.g., *resource consumption*, lower values are preferred, whereas for others, as e.g., *profit*, higher values are preferred. We refer to the criteria of these two cases as *lower best* and *higher best* criteria.

Criteria can also be classified based on the direction in which their values are altered by the actions of a planning problem. The criteria, the values of which change in a single direction, are called *monotonic* (*increasing* or *decreasing*, based on the specific direction), while the others are called *non-monotonic*. The monotonic criteria in particular can also be divided into *worsening monotonic criteria*, the values of which change towards their worst values, and into *improving monotonic criteria*, the values of which change towards their best values. The former are usually cost criteria, whereas the latter are usually profit ones.

In decision making [19,23,35], criteria can be organized in hierarchies. For the lowest-level criteria, called *basic criteria*, a method of measurement is defined in order to assign values to them. For the highest-level criteria, called *compound criteria*, an aggregation method is defined, so that the values from the basic criteria can be combined and give an overall value for the evaluated object, e.g., the plan. There are several aggregation methods, such as utility-based methods, outranking methods, analytical hierarchy process, etc., each being suitable for different types of evaluation problems. MO-GRT adopts the Weighted Average Sum method (WAS), which is a linear multi-attribute value function, suitable for multi-attribute and multiobjective deterministic problems with arithmetic criteria and large numbers of evaluated entities, and results in a cardinal ranking among the alternatives. For the correct application of WAS, weights have to be assigned to the criteria, representing the relative preferences of the evaluator with respect to each criterion.

An example of a criteria hierarchy for a transportation logistics problem is shown in Fig. 3. This hierarchy consists of two levels only; however, the basic criteria can be further analyzed to produce a deeper hierarchy. For example, the criterion of *safety* can be decomposed into *road condition*, *truck condition*, *area security*, *driver experience*, etc.

The criterion *length* is considered separately from the criterion of *duration*, since the former refers to the number of actions in a plan, while the latter refers to the cumulative duration of their sequential execution. Actually, the length of a plan reflects the difficulty in constructing it, while the duration of a plan is related to the difficulty in executing it.

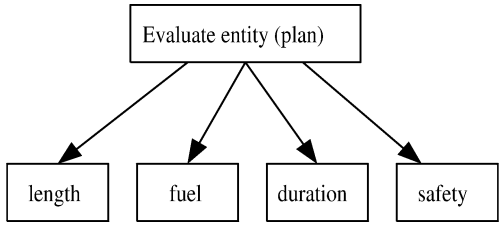

Fig. 3. A simple criteria hierarchy for the logistics domain.

However, in problems where all actions have equal durations, both criteria are equivalent and one of them should be omitted.

A scale is assigned to each basic criterion, including the indication whether higher or lower values are preferred. For example, the scale of *plan duration* for a specific problem may be the interval (20, 40) and lower values are preferred. This does not necessarily mean that plans with duration below 20 or above 40 time units will be pruned; it rather means that these plans will be evaluated as if they had a duration of 20 or 40 time units, respectively. The reason for setting scales for the basic criteria is twofold: Firstly, it prevents us from having extremely good plans, especially for the criteria the values of which can change towards their best values. Secondly, it allows us to normalize the values of all criteria in a common scale, in order to aggregate them.

Scales also play another role: Through the adoption of the WAS method for the aggregation of the values of the basic criteria, we implicitly considered that the evaluation function is linear. However, this assumption is too strong to be true in the whole real numbers interval. Thus, setting scales restricts this linearity within the scales only, which is a more real assumption.

In many cases, the scale bounds (or at least one of the two scale bounds) are hard. For example, we might not accept a plan with duration greater than 40 time units. In the presence of hard bounds, MO-GRT prunes the plans that are definitely out of the bounds and gives low priority to plans estimated to be out of the bounds. In this paper, we use brackets to denote soft bounds and square brackets to denote hard bounds, e.g., (20, 40].

Note that there is no reason to set a hard bound in the lower values of a monotonic increasing criterion or in the higher values of a monotonic decreasing one. Moreover, we do not consider the case of setting hard bounds in the best values of a criterion, although even this case cannot be de facto disregarded.

The definition of scales affects the results of the evaluation process significantly. For example, if all the produced plans are of a duration between 20 and 40 time units and we have set the scale of the criterion *duration* to the interval (0, 1000), then all plans will be considered as near optimal and will get about the same score with respect to *duration*. On the other hand, if we have set the scale of this criterion to the interval (20, 25), a plethora of plans with a duration of more than 25 time units will be considered as worst plans and will get exactly the same score. Deciding a criterion's scale is a critical issue and requires careful analysis of the evaluator's preferences.

5.2. The multiobjective heuristic function

The most difficult part of the evaluation process is the estimation of the cost of achieving the goals from each state of the state-space. MO-GRT extends the heuristic function of the single-objective GRT planner, by assigning each fact p cost-vectors of the form:

$$\langle \mathit{Length}, C_1, C_2, \ldots, C_N \rangle$$

which estimate the cost of the various paths that achieve p from the goals (N stands for the number of basic criteria, whereas the criterion *length* is considered separately). The set

of cost-vectors assigned to a fact p is denoted with $\mathbf{V}(p)$ and is computed by the following recursive formula, which generalizes formula (8):

$$\mathbf{V}(p) = \mathit{non\_dominated} \begin{cases} \langle 0, 0, \ldots, 0 \rangle, & \text{if } p \in \mathit{Goals}. \\ \textbf{AGGREGATE}(\mathit{Pre}(a')) + \langle 1, r_1, \ldots, r_N \rangle, & \\ \qquad \text{for each inverted action } a', & \\ \qquad \text{so that } p \in \mathit{Add}(a'). & \\ \qquad r'_i s,\ i = 1, \ldots, N, \text{ denote the} & \\ \qquad \text{contribution of } a' & \\ \qquad \text{to the basic criteria.} & \\ \infty, & \text{otherwise} \end{cases} \tag{10}$$

In (10), the prefix operator *non_dominated* operates on sets of cost-vectors and returns the subset of non-dominated ones. Function **AGGREGATE** is identical with that of the single-objective GRT planner, except for the fact that it aggregates cost-vectors instead of single values.

As mentioned in Section 4, a fact p can be achieved from the goals in several ways. In case of a single criterion, only a single path with the lowest possible cost is considered. However, when multiple criteria are taken into account, the notion of the lowest cost is vague. A path may be best in one criterion, whereas other paths may be best in the rest of the criteria. Thus, a set of cost-vectors must be assigned to each fact. A cost-vector is worth to be considered only if it is non-dominated by another cost-vector, otherwise it is rejected. This is a consequence of the fact that the WAS function adopted by MO-GRT has a linear form and hence it satisfies the monotonicity property.

If a set of vectors is assigned to each fact p, then function **AGGREGATE** has to be applied to any combination of the different vectors of its arguments resulting in a set of cost-vectors, i.e., a cost-vector for each different combination. Note that a different set of related facts is assigned to each cost-vector, depending on the specific sequence of actions that established this cost-vector.

5.2.1. Complexity problems

MO-GRT faces the risk of combinatorial explosion both in memory and in time requirements. Memory requirements concern the space needed to store the non-dominated cost-vectors assigned to each fact. Even for a few criteria, the average number of cost-vectors per fact may be large. On the other hand, time requirements concern the application of function **AGGREGATE** to all combinations of the cost-vectors of its arguments.

Suppose $\overline{V}$ is the average number of cost-vectors per fact, $\overline{P}$ is the average number of precondition facts per action and $\overline{F}$ is the average number of facts per state. In this case, for the application of an inverted action in the heuristic construction phase, $\overline{V}^{\overline{P}}$ combinations should be considered on average. On the other hand, when estimating the cost of reaching the goals from a state of the search phase, only the best cost-vectors need to be considered first, according to the evaluation function. However, in case the resulting vector exceeds the hard bounds of some criteria, alternative combinations of the cost-vectors of the state's facts have to be considered. In the worst case, these combinations are $\overline{V}^{\overline{F}}$. For a better notion of these numbers, let us consider that $\overline{V} = 5$, $\overline{P} = 3$ and $\overline{F} = 30$, which are some usual values. In this case, we have an average of $\overline{V}^{\overline{P}} = 5^3 = 125$ applications of function

AGGREGATE for each applied inverted action during the heuristic construction phase and $\overline{V}^{\overline{F}} = 5^{30}$ applications of function **AGGREGATE** in the worst case for each state during the search phase.

In order to preclude potential complexity problems, MO-GRT adopts a looser selection method in storing and combining cost-vectors. We refer to this method as the *relaxed dominance pruning heuristic*. This method reduces the number of cost-vectors retained for each fact to the following ones:

(1) The best cost-vector, according to the evaluation function.
(2) For each worsening monotonic criterion, the cost-vector with the best value in this criterion is also retained.
(3) For each improving monotonic criterion, the cost-vector with the best combined value in the rest of the criteria, is also retained.
(4) For the non-monotonic criteria, both case 2 and case 3 are applied, thus two additional cost-vectors are retained for each one of them.

The rationale underlying the selection of the above cost-vectors is the following: The best cost-vector of each fact is retained, since combining these cost-vectors of a set of facts (e.g., the preconditions of an inverted action or the facts of a state) may result to the best combined cost-vector for these facts, according to Corollary 4, which holds for the WAS function. However, this combined vector may probably exceed the scales for some criteria, so alternative vectors of the facts have to be tried.

For the case worst bounds (either hard or soft) are violated, the relaxed dominance method retains the vectors that have best values and are within the bounds (case 2). On the other hand, violating the best soft bound of a criterion does not contribute positively to the overall value of the cost-vector, since the vector is evaluated as if it had the value of the bound. Thus, an attempt is made to find a new cost-vector that optimizes the overall value, even if it is still out of the bound for the specific criterion, rather than to find a new cost-vector inside the bound. This is the reason why in case 3 the vector that optimizes the combined value of the other criteria is retained.

Suppose now that we have N basic criteria (*length* being excluded), N_1 of which are monotonic and N_2 of which are non-monotonic. In this case, the maximum number of cost-vectors that will be retained for each fact would be:

$$1 + N_1 + 1 + 2 \cdot N_2$$

where 1 stands for the best vector, $N_1 + 1$ stands for the one additional vector retained for each monotonic criterion (*length* being included) and $2 \cdot N + 2$ stands for the two additional vectors retained for each non-monotonic criterion.

In some cases, it is also possible to retain cost-vectors that exceed hard bounds of the scales. The strategy adopted is the following:

- The first cost-vector for each fact is retained, even if it exceeds some hard bounds.
- A new cost-vector that exceeds some hard bounds is rejected, provided that there is an existing cost-vector, which does not violate any hard bound to a greater extent than the new one.

According to the above rules, all the non-dominated cost-vectors that exceed some hard bounds are retained. However, as soon as a new cost-vector that does not violate any hard bound is produced, all the existing violating cost-vectors are eliminated.

Note that the criteria bounds used in the heuristic construction phase are not the original ones. Suppose a criterion c has a scale (L_c, R_c) and an initial amount of $Init_c$. In this case, the scale used for this criterion in the heuristic construction phase is $(L_c - Init_c, R_c - Init_c)$. This is a consequence of the assignment of zero cost-vectors to the goal facts and this is because in the construction of the heuristic function we are only interested in the remaining cost of achieving the goals from any intermediate state and not in the cost paid for reaching the intermediate state from the initial one.

5.2.2. State evaluation

In order to estimate the cost of achieving the goals from any intermediate state, MO-GRT assigns a cost-vector to the state, by applying function **AGGREGATE** to the cost-vectors of the state facts. Certainly, there are many cost-vectors that can be produced, which represent the alternative paths in which the goals can be achieved from the current state. However, MO-GRT computes a single cost-vector only, which is considered to correspond to the "best" path.

Firstly, MO-GRT considers the best cost-vectors of the state-facts. In case the resulting vector does not violate any bound, it is assigned to the state. Note that, in this case, new scales reflecting the current resource availability are considered, rather than the original ones. Suppose the original scale of a criterion is (L_c, R_c) and the cost of the current plan with respect to this criterion is c. In this case, the scale of this criterion for the specific state is considered to be $(L_c - c, R_c - c)$.

In case the state cost-vector resulting from the best cost-vectors of the state facts violates some bounds, the alternative cost-vectors of the state-facts are attempted, giving priority to the vectors that reduce the extent of the violations and then to the vectors that improve the value of the resulting state cost-vector. As soon as a state cost-vector that does not violate any bound is produced, the process stops and this vector is assigned to the state. However, in the worst case, this process would go on until all the combinations of the alternative cost-vectors of the state-facts have been considered without producing a non-violating state cost-vector. In order to overcome the potential complexity problem in similar situations, MO-GRT adopts a greedy hill-climbing like approach: It computes an overall violation for the state cost-vector and then, it allows to substitute a fact cost-vector with an alternative one only in case the new state cost-vector has a lower overall violation. This process continues until no improving fact cost-vector can be found. Of course, the approach adopted by MO-GRT is heuristic, so there is still a possibility not to find a non-violating state cost-vector, although such one exists.

5.3. Plan evaluation

The criteria hierarchy is used to evaluate the states of the state-space. These must be evaluated both for the known accumulated cost of the past plan and the estimated cost of the remaining plan towards the goals, based on the heuristic estimations. Thus, the criteria hierarchy has to be applied twice and both values have to be combined. The only

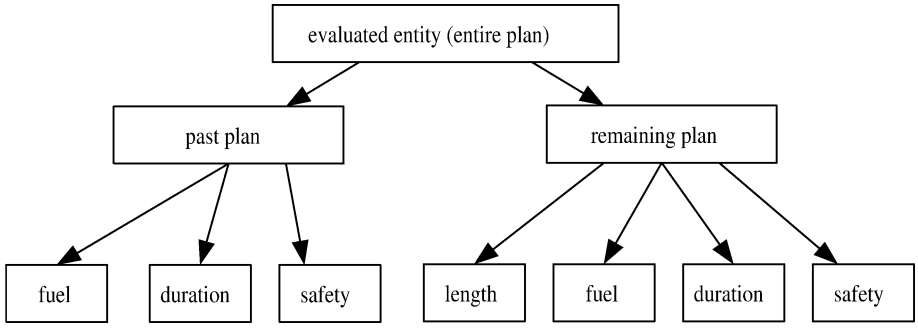

Fig. 4. The integrated criteria hierarchy for the evaluation of the states.

modification is that the criterion *length* is of no interest for the past plan, except for the case where this criterion reflects the duration of the plan. Fig. 4 shows the integrated criteria hierarchy for the entire plan, based on the simple criteria hierarchy of Fig. 3.

The values assigned to the two top-level criteria, i.e., the past plan and the remaining plan, have to be combined using weights. The integrated function used for the evaluation of the states is formed as:

$$f(S) = W_p \cdot E\big(\mathbf{g}(S)\big) + W_r \cdot E\big(\mathbf{v}(S)\big), \quad W_p + W_r = 1,\ W_p \cdot W_r \geqslant 0 \tag{11}$$

where S is the evaluated state, $\mathbf{g}(S)$ is the cost-vector of the past plan, $\mathbf{v}(S)$ is the cost-vector of the remaining plan that has been assigned to the state, W_p is the relative weight of the past plan and W_r is the relative weight of the remaining plan. For $W_p = W_r = 0.5$, the search behaves as the original A* strategy, for $W_p = 1$, the search behaves as a breadth first optimal strategy, whereas for $W_r = 1$ the search behaves as the greedy best-first strategy.

Note, however, that the precise behavior of the heuristic function depends also on the weights assigned to the lower-level criteria. Thus, the words "optimal" and "greedy" do not refer to the length of the plan, as is usually the case for single-objective planners; they refer to the weighted accumulation of the values for all criteria.

A crucial point is the treatment of the states, to which an estimated cost-vector of the remaining plan that violates some hard bounds has been assigned. There are two alternative approaches to handle these states, depending on whether the heuristic function is admissible or not. In case of an admissible function and without the use of the relaxed dominance heuristic pruning method, these states can safely be pruned; otherwise, they have to be retained. Since the heuristic of MO-GRT is not admissible, the planner retains all states of the frontier set, however it penalizes the states that violate a hard bound by twice the amount of the violation.

This section ends with the application of the notion of domination to the states. The single-objective GRT keeps a closed list of visited states, in order to avoid re-visiting them. In the case of MO-GRT, this closed list has to be extended, in order to store the non-dominated cost-vectors of the several visits in the state. Now, a revisited state is only pruned in case the vector of a previous visit dominates the vector of the new one in all basic criteria.

5.4. Resources and achievability analysis

Prior to the heuristic construction phase, GRT performs an achievability analysis of the problem facts. This analysis is performed in a way quite similar to the way GRAPHPLAN-like planners construct their planning graph [2]. During this, the actions of the problem are repeatedly applied to the initial state, until no more facts can be reached. The purpose of this analysis is twofold: Firstly, facts that cannot be achieved from the initial state are detected and removed from the subsequent phases (however, it is not certain that all non-achievable facts will be detected). Secondly, mutual exclusion relations between the problem facts are identified. These relations are used by GRT in the detection and completion of incomplete goal states [29].

MO-GRT could use exactly the same techniques, as GRT, by simply ignoring the application costs of the actions. However, MO-GRT enhances the planning graph, by taking into account hard bounded monotonic criteria and computing a lower bound of the requirements of resources needed to achieve each fact of the problem. In case these requirements exceed the available resources for a fact, the fact is considered unachievable and is not considered in the heuristic construction phase and in the search phase.

In the enhanced planning graph, a cost-vector is assigned to each fact, where the dimensions of this vector correspond to the bounded monotonic criteria. This vector expresses the least resource amount needed to achieve the fact from the initial state. Similarly, a vector is assigned to each action of a problem; this vector expresses a lower bound of the cost needed to apply the action for the first time. The creation of the planning graph is based on the following rules:

- The vector assigned to the initial state facts reflects the initial resource availability.
- The vector of an action is constructed by adding the worst value of each dimension of the action's preconditions' vectors to the application cost of the action for this dimension.
- An action can be applied, if its vector is within the resource boundaries for all its dimensions.
- The vector of each action is assigned to the facts that are firstly achieved by that action.
- If a fact is re-achieved by an action, the new vector of the fact is constructed from the best values of its previous vector and the action's vector, for each dimension.
- Each time the vector of an action's precondition changes, the vector of the action is recomputed and the action is re-considered for application.

As far as the mutual exclusion relations between facts are concerned, these are computed in exactly the same way as in GRAPHPLAN. It is easy to prove that the use of the above rules for the construction of a planning graph, does not result neither in an inability to achieve an achievable fact nor in a false consideration of two facts as mutually exclusive. However, there are still unachievable facts that remain undetected. A more elaborative construction of the planning graph would demand retaining multiple non-dominated cost-vectors for the actions and for the facts, as well as separate sets of mutual exclusion relations for each cost-vector, in a way similar to the MO-GRT heuristic construction phase. This approach

has not yet been implemented in MO-GRT and it is an interesting direction for future investigation.

5.5. Example

This section demonstrates how MO-GRT evaluates states. A simple example is selected, where no facts are related to each other and the goals constitute a complete state description.

Suppose there are three cities, *city1*, *city2* and *city3*, and a package that must be moved from *city1* to *city3*. The package can be moved between two cities either by truck or by plane. The cities *city1* and *city3* are not directly connected, so the package must firstly be moved to *city2* and then to *city3*. There are two action schemas,

$$O_1 = (\textit{move-by-truck}\ ?\textit{Package}\ ?\textit{City1}\ ?\textit{City2}) \quad \text{and}$$
$$O_2 = (\textit{move-by-plane}\ ?\textit{Package}?\textit{City1}\ ?\textit{City2})$$

where variable names begin with a question mark. Both schemas have identical preconditions and effect lists, which are defined as follows:

$$\begin{aligned} \textit{Pre}(O_1) = \textit{Pre}(O_2) = \big((&\textit{package}\ ?\textit{Package})\ (\textit{city}\ ?\textit{City1})\ (\textit{city}?\textit{City2}) \\ &(\textit{connected}\ ?\textit{City1}\ ?\textit{City2})(\textit{at}\ ?\textit{Package}\ ?\textit{City1})\big) \\ \textit{Del}(O_1) = \textit{Del}(O_2) = \big((&\textit{at}\ ?\textit{Package}\ ?\textit{City1})\big) \\ \textit{Add}(O_1) = \textit{Add}(O_2) = \big((&\textit{at}\ ?\textit{Package}\ ?\textit{City2})\big) \end{aligned}$$

Note that the above action schemas are general definitions, which can be grounded in several ways, resulting in many actions.

Suppose that these two action schemas have different application cost and duration. We will use the notation $\langle \textit{length}, \textit{cost}, \textit{duration} \rangle$ to denote cost-vectors. Assume that action schema O_1 has an application cost of $\langle 1, 10, 30 \rangle$ and action schema O_2 has an application cost of $\langle 1, 40, 10 \rangle$.

For all criteria the lowest values are considered best. Moreover all criteria monotonically increase their values towards their worst values (worsening monotonic criteria). So, according to the relaxed dominance pruning heuristic, the cost-vectors with either the best combined value or the lowest value in at least one criterion will be retained. Table 2 presents the weights and the scales for the three criteria of this problem. Note that for this problem the upper bounds of the scales are considered hard (the types of the left bounds are of no importance, since the criteria are monotonic increasing).

The initial state and the goals are defined as follows:

$$\begin{aligned} \textit{Initial} = \big((&\textit{city city1})\ (\textit{city city2})\ (\textit{city city3})\ (\textit{package package1}) \\ &(\textit{connected city1 city2})\ (\textit{connected city2 city1})\ (\textit{connected city2 city3}) \\ &(\textit{connected city3 city2})\ (\textit{at package1 city1})\big) \\ \textit{Goals} = \big((&\textit{at package1 city3})\big) \end{aligned}$$

Table 2
The weights and the scales of the criteria

Criterion	Weight	Scale
Length	3	(0, 5]
Cost	2	(0, 200]
Duration	1	(0, 160]

In the heuristic construction phase, only the dynamic facts, i.e., the facts that can be added or deleted by an action, are considered. This problem has three dynamic facts, i.e., (*at package1 city1*), (*at package1 city2*) and (*at package1 city3*).

The inverted actions are used for the assignment of cost-vectors. In this simple problem, the set of inverted actions is identical to the set of normal ones, so we will use the normal actions for simplicity. These are the following eight ground actions:

a_1 = (*move-by-truck package1 city1 city2*)

a_2 = (*move-by-truck package1 city2 city1*)

a_3 = (*move-by-truck package1 city2 city3*)

a_4 = (*move-by-truck package1 city3 city2*)

a_5 = (*move-by-plane package1 city1 city2*)

a_6 = (*move-by-plane package1 city2 city1*)

a_7 = (*move-by-plane package1 city2 city3*)

a_8 = (*move-by-plane package1 city3 city2*)

The vector $\langle 0, 0, 0\rangle$ is assigned to the fact (*at package1 city3*), since this is a goal fact. We see that fact (*at package1 city2*) can be achieved from the goals by two actions, the a_4 and a_8. Using a_4 results in the cost-vector $\langle 1, 10, 30\rangle$, whereas using a_8 results in the cost-vector $\langle 1, 40, 10\rangle$. These two vectors are evaluated, based on the weights and scales of the criteria (the scales are used for the normalization of values). The normalized scales are the same for all criteria, e.g., the interval (0, 100), and lowest values are considered best. Consider the vector $\langle 1, 10, 30\rangle$. After the normalization, this vector is transformed to $\langle 20, 5, 18.75\rangle$. Then, the dimensions of the vector are merged in a single value, with the use of weights; this results in an overall value of $(3 \cdot 20 + 2 \cdot 5 + 1 \cdot 18.75)/6 = 14.79$. Similarly, the vector $\langle 1, 40, 10\rangle$ is normalized to the vector $\langle 20, 20, 6.25\rangle$, resulting in an overall value of 17.70. Since lower values are preferred, the vector $\langle 1, 10, 30\rangle$ is selected as the best cost-vector for the fact (*at package1 city2*). However, the cost-vector $\langle 1, 40, 10\rangle$ is also retained, since it has the best value in criterion *duration*.

In the above computations, none of the cost-vectors violated the bounds of the criteria. Note also that the scales used in the heuristic construction phase are the original ones, since the value of all criteria in the initial state is zero.

Next, we see that the fact (*at package1 city1*) can be achieved by two actions, i.e., a_2 and a_6. These actions have as precondition the fact (*at package1 city2*), which has two cost-vectors, thus each action produces two cost-vectors. In particular, action a_2

produces the vectors $\langle 2, 20, 60\rangle$ and $\langle 2, 50, 40\rangle$, whereas action a_6 produces the cost-vectors $\langle 2, 80, 20\rangle$ and $\langle 2, 50, 40\rangle$. The three cost-vectors, $\langle 2, 20, 60\rangle$, $\langle 2, 50, 40\rangle$ and $\langle 2, 80, 20\rangle$ are normalized to the vectors $\langle 40, 10, 37.5\rangle$, $\langle 40, 25, 25\rangle$ and $\langle 40, 40, 12.5\rangle$, which are evaluated to the combined values 29.58, 32.5 and 35.42, respectively. The cost-vector $\langle 2, 20, 60\rangle$ is considered best, the cost-vector $\langle 2, 50, 40\rangle$ is rejected since it has not a best value in a single criterion (according to the relaxed dominance pruning heuristic), whereas the cost-vector $\langle 2, 80, 20\rangle$ is retained since it has the best value in the criterion of *duration*. The presentation of the heuristic construction phase ends here, since there are no other facts that can be achieved in a better way (however the process actually stops when all the inverted actions have been applied at least once).

Suppose now that there is another package, e.g., *package2*, which is also initially located in *city1* and has to be moved to *city3*; so, the same cost-vectors are assigned to the facts (*at package2 city1*), (*at package2 city2*) and (*at package2 city3*), as for object *package1*. Now the initial state becomes:

$$\begin{aligned} \mathit{Initial} = \big(&(\mathit{city\ city1})(\mathit{city\ city2})(\mathit{city\ city3})(\mathit{package\ package1})\\ &(\mathit{package\ package2})(\mathit{connected\ city1\ city2})(\mathit{connected\ city2\ city1})\\ &(\mathit{connected\ city2\ city3})(\mathit{connected\ city3\ city2})(\mathit{at\ package1\ city1})\\ &(\mathit{at\ package2\ city1})\big) \end{aligned}$$

Next, an overall value is computed and assigned to the initial state, in the following way: For each dynamic fact of the initial state, its best cost-vector is selected, provided that the resulting combined vector does not violate the scales. In this case, the best cost-vector for each dynamic fact of the initial state is $\langle 2, 20, 60\rangle$, resulting in the combined cost-vector $\langle 4, 40, 120\rangle$, which is normalized to the cost-vector $\langle 80, 20, 75\rangle$ and results in the evaluation value of 59.17. The combined cost-vector does not violate any bound, so there is no reason to look in the alternative cost-vectors of the initial state facts.

This value is assigned to the remaining plan of the initial state. As for the past plan of the initial state, both its vector and its normalized one for this problem are $\langle 0, 0, 0\rangle$, resulting in an overall value of 0. Both overall values have to be combined with the use of weights. Suppose that the weight is 1 for the past plan and 3 for the remaining plan; in this case, both overall values are combined to the value $(1 \cdot 0 + 3 \cdot 59.16)/4 = 44.37$. This is the score of the initial state. The described procedure is used to evaluate the states of the state space and those with lower scores are selected for expansion.

6. Performance measurements

This section examines the role that the criteria, their weights and scales play in the planning process, i.e., how they affect the planning time and the quality of the resulting plans. This is performed through an adequate number of experiments in an enhanced *logistics*-type domain.[3]

[3] All files needed for the reproduction of the experiments, i.e., the planner (executable file and source code), problem files, script files and the data files are available at the URL: http://macedonia.uom.gr/∼yrefanid/GRT/.

Table 3
Application cost and duration for the actions of the *logistics*^{MO} domain

Actions	Cost	Duration
Loading/unloading any truck	2	1
Loading/unloading any train	2	1
Loading/unloading any plane	3	2
Moving a truck	10	10
Moving a train	20	100
Flying a plane	50	10

6.1. The *logistics*^{MO} domain

The original *logistics* domain [34] consists of several locations in several cities. One location in each city is labeled as airport. Each city has one or more trucks, which can move between its locations. There are also several planes that can fly between the airports. Finally, there are some packages that must be transported from their initial to their final locations by trucks and planes.

In the above description, there is a single means of transportation to transfer an object between two cities, i.e., an airplane, since trucks are only used for intracity transportations. In order to measure the effectiveness of MO-GRT, we have extended this description with trains, which can only perform transportations between different cities and we have labeled one location in each city as a railway station. We call this extended logistics domain *logistics*^{MO}.

The new domain has three new actions, referring to the loading, unloading and moving of a train. Moreover, two predicates, namely *train* and *station*, have been introduced: they describe an object as train and railway station, respectively. We have also introduced the criteria of *cost* and *duration* and we have assigned an application cost and duration to all domain actions schemas (Table 3). Certainly, lower values are preferable.

As it results from Table 3, all criteria (including *length*) are monotonically increasing. Besides, since lower values are preferred, the criteria are also worsening criteria. However, the adoption of a single type of criteria for the experiments does not restrict the generality of the results, since MO-GRT deals with all types of criteria equally.

6.2. Problem definition

As a starting point, we used the STRIPS untyped *logistics* problem set of the AIPS-00 planning competition [1], comprising 28 problems. In every problem of this distribution, all cities have two locations, one of which is the airport. For the *logistics*^{MO} domain, we labeled the non-airport location of each city as railway station. Furthermore, we added a train to each problem, initially located in the railway station of the first city. Note, finally, that the initial values of all criteria are considered to be zero.

To apply MO-GRT, we must set the criteria scales, which are not identical in all problems. In order to render the reproduction of the experiments feasible, we used an "algorithmic" way of setting these scales. Thus, these were based on the number of packages that had to be moved inside one city, or to a different one. We omitted packages

Table 4
Scales for the three criteria

Criteria	Left bound	Right bound
Length	(Right bound)/4	$4 \cdot P_1 + 12 \cdot P_2$
Cost	(Right bound)/8	$24 \cdot P_1 + 154 \cdot P_2$
Duration	(Right bound)/8	$22 \cdot P_1 + 246 \cdot P_2$

P_1 = Packages that must be transferred inside the same city.
P_2 = Packages that must be transferred to a different city.

Table 5
Weights used in various experiments

Experiment	Weights				
	Past plan	Remaining plan	Length	Cost	Duration
A	1	3	3	1	1
B	1	1	3	1	1
C	1	2	3	1	1
D	0	1	3	1	1
E	1	3	1	1	1
F	1	3	2	1	1
G	1	3	5	1	1
H	1	3	10	1	1
I	1	3	3	3	1
J	1	3	3	10	1
K	1	3	3	1	3
L	1	3	3	1	10

that were not referenced within the goals, as well as packages with identical initial and goal positions. Table 4 shows the formulas used to set the scales for all criteria.

The rationale of the above formulas is the following: As for the right bound, the formulas describe the worst cases, i.e., cases where the packages are transferred separately and by the worst means of transportation wrt each criterion (e.g., an airplane is considered wrt cost), while a means of transportation is never in place for transfer and it must be moved from another position. On the other hand, the decision for the left bound was based on our experience with the problems of the *logistics*^{MO} domain. The intention was to have all the solutions between the two bounds and to have a sensible distance between the left bound and the obtained values for all criteria.

We performed 12 experiments, denoted with the letters A to L. Each experiment included running the planner for all 28 problems of our *logistics*^{MO} problem set, using specific weights. Table 5 summarizes the weights used in these experiments.

Note that in the *logistics*^{MO} domain, the criterion *length* does not clearly favor any of the other two criteria, since the average actions needed to perform a transportation are the same, whether a plane or a train is used. However, this would not be the case if, for example, loading a package in a plane would require more than one actions. In general, the criterion *length* is usually positively related to some criteria and negatively related to some others.

6.3. Experimental results

MO-GRT has been implemented in C++. The measurements were taken on a SUN Enterprise 3000 machine running at 167 MHz, with 256 MB main memory under Solaris 2.5.1 OS. We set a CPU time limit equal to 5 minutes for each problem. Some problems were not solved due to memory limitations or due to the requirement for more processing time.

Next we compare several groups of experiments, where each group includes experiments that only differ in a single weight. Experiment A is included in all groups and serves as a reference. When comparing two experiments, e.g., experiment X to A, the following metrics are used:

$$
\begin{aligned}
m_{\text{solved}} &= \frac{solved(X) - solved(A)}{solved(A)} \\
m_{\text{time}} &= average_i\left(\frac{time(X_i) - time(A_i)}{time(A_i)}\right) \\
m_{\text{length}} &= average_i\left(\frac{length(X_i) - length(A_i)}{length(A_i)}\right) \\
m_{\text{cost}} &= average_i\left(\frac{cost(X_i) - cost(A_i)}{cost(A_i)}\right) \\
m_{\text{duration}} &= average_i\left(\frac{duration(X_i) - duration(A_i)}{duration(A_i)}\right)
\end{aligned}
\tag{12}
$$

where:

- $solved(Z)$ is the number of problems solved in experiment Z,
- $time(Z_i)$ is the time needed to solve problem i in experiment Z,
- $length(Z_i)$ is the length of the solution to the problem i in experiment Z,
- $cost(Z_i)$ is the cost of the solution to the problem i in experiment Z,
- $duration(Z_i)$ is the duration of the solution to the problem i in experiment Z,
- Z is an experiment (in this case X or A).

Note that the averages are computed in the problems solved in both experiments (X and A). There are two reasons why we use averages on a large number of problems, instead of comparing on specific problems. The first one concerns the large number of problems, the huge number of interesting variations in weights and scales, and the five metrics of interest, which makes the detailed presentation of the results in all cases impossible, due to the limited space of the paper. The second reason is that the impact of changing a weight or a scale may be negligible or even contradictory for a specific problem, due to the heuristic nature of the proposed technique; however the average results are always intuitive. Table 6 presents the absolute values in all 28 problems for experiment A, which are used as a basis of comparison in the subsequent sections.

Table 6
Detailed results for experiment A (solution time in msecs)

Problem	Solution time	Length	Cost	Duration	Problem	Solution time	Length	Cost	Duration
Logistics-4-0	170	13	126	238	Logistics-9-0	540	26	296	480
Logistics-4-1	180	15	162	152	Logistics-9-1	410	22	186	170
Logistics-4-2	130	12	164	158	Logistics-10-0	930	29	308	490
Logistics-5-0	190	19	170	156	Logistics-10-1	1350	39	412	628
Logistics-5-1	200	12	76	228	Logistics-11-0	1190	36	440	618
Logistics-5-2	120	8	32	26	Logistics-11-1	1520	47	486	656
Logistics-6-0	230	22	112	256	Logistics-12-0	1120	35	326	502
Logistics-6-1	130	9	52	126	Logistics-12-1	1250	42	420	712
Logistics-6-2	240	21	178	162	Logistics-13-0	2530	53	648	774
Logistics-6-9	170	15	132	242	Logistics-13-1	2070	43	504	636
Logistics-7-0	490	31	346	406	Logistics-14-0	3330	47	422	832
Logistics-7-1	550	33	308	406	Logistics-14-1	2610	49	448	734
Logistics-8-0	610	26	244	388	Logistics-15-0	4510	64	686	724
Logistics-8-1	800	29	284	384	Logistics-15-1	3150	61	700	698

Table 7
Results for several combinations of past and remaining plan weights

Experiment	m_{solved}	m_{time}	m_{length}	m_{cost}	m_{duration}
B (past = 1, remaining = 1)	−35.71%	913.68%	−1.01%	−0.88%	−8.68%
C (past = 1, remaining = 2)	0.00%	52.64%	−0.72%	−3.31%	−0.47%
D (past = 0, remaining = 1)	0.00%	−13.92%	0.99%	2.91%	4.75%

In the following sub-sections we examine the effect of the weights of the past and the remaining plans, as well as the weights and scales of all criteria, on the planning process. We close this section by comparing MO-GRT to GRT in the same problems.

6.3.1. Varying the weights of the past and the remaining plan

In this section, we compare experiments B, C and D to experiment A. We investigate the effect of different combinations of the past and the remaining plans weights on the overall planning process. Actually, what is of interest is the ratio between these weights. Table 7 shows the results.

Let us first interpret the results in Table 7, e.g., for experiment B: Running MO-GRT in the 28 problems of our problem set, with the weights of experiment B, results in solving 35.71% less problems than when running MO-GRT with the weights of experiment A and in the time limit of 5 minutes. Besides, in problems solved in both experiments, the case of experiment B needed 913.68% more time on average to solve them, it produced plans that were 1.01% shorter and they had 0.88% smaller cost and 8.68% smaller duration on average than in case of experiment A.

Table 7 shows that as the ratio between the weights of the remaining plan and the past plan increases, the planner reaches faster a solution. Furthermore, in experiment B, where we had the lowest value of this ratio, 35% of the problems (the largest ones) were not

Table 8
Results for several weights of the criterion *length*

Experiment	m_{solved}	m_{time}	m_{length}	m_{cost}	m_{duration}
E (length = 1)	0.00%	130.19%	3.76%	1.03%	−2.05%
F (length = 2)	0.00%	14.32%	1.88%	−0.61%	1.14%
G (length = 5)	0.00%	−7.38%	−0.49%	0.89%	−1.16%
H (length = 10)	0.00%	−12.42%	−0.40%	1.29%	−1.09%

Table 9
Results for various weights of the criteria *cost* and *duration*

Experiment	m_{solved}	m_{time}	m_{length}	m_{cost}	m_{duration}
I (cost = 3)	0.00%	13.72%	−0.27%	−9.77%	7.06%
J (cost = 10)	−17.86%	1509.18%	0.68%	−14.55%	9.16%
K (duration = 3)	0.00%	64.74%	6.93%	9.89%	−9.26%
L (duration = 10)	−21.43%	1161.74%	22.53%	32.29%	−17.72%

solved. With regard to the three criteria, as the above ratio decreases, the produced plans generally become better. However, the degree of this effect is different for the three criteria.

6.3.2. Varying the weight of the criterion length

In this section, we compare experiments E, F, G and H to experiment A. We investigate the effect of the weight of the criterion *length* on the overall planning process. Table 8 shows the results.

The above results show the effect of the weight of the criterion *length* on the total solution time. As the weight of this criterion increases, the planner reaches faster a solution and finds slightly shorter plans; on the other hand, as this weight decreases, the planner delays and finds slightly longer plans. The effect of this weight on the plan cost and duration is neither significant nor consistent. This is explained by the fact that in the *logistics*$^{\text{MO}}$ domain the plan length is not competitive either to the plan cost or the plan duration.

6.3.3. Varying the weights of the criteria cost and duration

In this section, we compare experiments I, J, K and L to experiment A. We investigate the effect of the weights of the criteria *cost* and *duration* on the overall planning process. Table 9 shows the results.

It is seen that for both the criterion *cost* (experiments I and J) and the criterion *duration* (experiments K and L), as the weight of each criterion increases, the resulting plans become better in terms of this criterion, while they worsen with respect to the rest of the criteria. It is also seen that an increase in the weight of the criterion *cost* does not significantly affect the length of the obtained plans, which does not occur in case of an increase in the weight of the criterion *duration*. The rationale of this observation is that most packages are initially located in railway stations and the same occurs in their goal positions. Thus, the demand for plans of lower duration favors the use of planes for transportation, which leads to longer plans as a side-effect.

A second observation is that as the weights of the criteria *cost* and *duration* increase, whereas the weight of the criterion of *length* remains the same, the solution time increases. Especially in case these weights become greater than the weight of the criterion *length* (experiments J and L), many problems cannot be solved within the time and memory limits. This result was expected, based on the results of Section 6.3.2, where the influence of varying the weight of the criterion *length* has been investigated.

6.3.4. Varying the scales of the criteria

In this section, we investigate the effect of the scales attached to the criteria on the overall planning process. Reusing the problems of experiment A as a reference, we constructed 12 variations, keeping the same weights and changing the scales. The new experiments are denoted with $A_{\text{criterion}\times M}$, where *criterion* is the criterion the scale of which has changed, and M is a positive number multiplying the width of the original scale. The new scale has the same center as the original one, but it is M times broader. For example, if the scale of *duration* in a problem of experiment A was initially (200, 300) and M was 2, the same scale in experiment $A_{\text{DURATION}\times 2}$ would be (150, 350). Both scales have the same center, i.e., 250, but the second one is two times broader than the first one. Note that in case the left bound becomes lower than 0, we set it to the value 0 and we shift the right bound accordingly. Table 10 shows the results.

The conclusion drawn from Table 10 is that a criterion scale affects significantly the quality of the resulting plan in terms of this criterion. The results show that as the scale of a criterion diversifies (broadens or shrinks) from a critical scale, the quality of the obtained plans reduces, in terms of this criterion, whereas it may increase in terms of the other criteria. For example, as we broaden or shrink the scale of the criterion *cost*, the cost of the obtained plans increases, while their duration decreases. On the other hand, as we broaden or shrink the scale of the criterion *duration*, the duration of the obtained plans increases. However, in this case we observe that when the scale broadens two times, the duration of the obtained plans decreases. This is an indication that a two-times broader scale is the critical scale for this criterion. We note finally that as the scales of the criteria *cost* and *duration* diversify from their critical scales, the planner reaches a solution significantly

Table 10
Results for various criteria scales

Experiment	m_{solved}	m_{time}	m_{length}	m_{cost}	m_{duration}
$A_{\text{COST}\times 2}$	0.00%	−5.27%	−0.08%	3.10%	−0.93%
$A_{\text{COST}\times 5}$	0.00%	−8.03%	1.95%	7.72%	−2.20%
$A_{\text{COST}\times 0.5}$	0.00%	−3.38%	2.02%	7.24%	−1.14%
$A_{\text{COST}\times 0.2}$	0.00%	−7.88%	1.85%	7.47%	−2.76%
$A_{\text{DURATION}\times 2}$	0.00%	−4.39%	0.38%	3.02%	−0.27%
$A_{\text{DURATION}\times 5}$	0.00%	−12.30%	0.01%	0.28%	5.24%
$A_{\text{DURATION}\times 0.5}$	0.00%	−2.31%	2.78%	1.50%	11.67%
$A_{\text{DURATION}\times 0.2}$	0.00%	−6.14%	1.55%	−1.82%	14.39%
$A_{\text{LENGTH}\times 2}$	0.00%	228.51%	2.03%	−1.60%	1.18%
$A_{\text{LENGTH}\times 5}$	−42.86%	2804.42%	3.89%	2.67%	−4.19%
$A_{\text{LENGTH}\times 0.5}$	0.00%	−7.84%	−0.22%	−0.93%	0.44%
$A_{\text{LENGTH}\times 0.2}$	0.00%	−7.88%	1.85%	7.47%	−2.76%

Table 11
Comparison between the single-objective GRT and the MO-GRT in experiment A

Experiment	m_{solved}	m_{time}	m_{length}	m_{cost}	m_{duration}
Single objective GRT	0.00%	−19.94%	0.16%	6.34%	−0.81%

faster. This is because as these two criteria lose their strength, the effect of the criterion *length* on the planning process becomes more significant.

As for the criterion *length*, we observe that as its scale broadens, the planner needs more time to find a solution and, for greater broadenings, many problems become unsolvable in the specified limits of time and memory. Inversely, when we shrink this scale, problems are solved faster. This leads us to the conclusion that the critical scale for the criterion *length* is significantly narrower than the originally selected one.

Finally, we would like to lay emphasis on the fact that there is no ideal scale for a criterion. One could identify the critical scale for a criterion and for a specific problem, by repeatedly running the planner on this problem using different scales for the criterion and observing the quality of the resulting plans in terms of this criterion. However, this is not the best scale to be used. The reason is that the best scale for each criterion and for a specific problem is a very subjective matter that depends only on the evaluator's preferences, i.e., the person who is interested in the solution plan, and may be very different from the critical one. Extracting the preferences of the evaluator (criteria hierarchy, weights and scales) is a difficult problem that has been thoroughly studied in the area of decision-making.

6.3.5. Comparing GRT and MO-GRT

We conclude our performance results by comparing the single-objective planner GRT to MO-GRT. GRT can be considered a special case of MO-GRT, if all criteria have zero weights, except for *length*, which has a weight equal to 1, and if the weight of the past plan is equal to 0, while the weight of the remaining plan is equal to 1.

Certainly, the conclusions drawn by the comparison depend on the weights and scales that will be used by MO-GRT. In order to retain a common reference in our experiments, we compare GRT to MO-GRT using again the parameters of experiment A (Table 11).

As Table 11 shows, GRT is approximately 20% faster than MO-GRT. Note that this acceleration is greater than all accelerations encountered in all experiments. As far as the other criteria are concerned, GRT found plans of approximately equal length, higher cost and lower duration.

7. Extending heuristic panning to support multiple criteria

This section briefly presents other single objective planners by stressing their specific features, and proposes techniques for adapting the multiobjective paradigm to them.

7.1. Single-objective heuristic planners

The recent development of the domain-independent heuristic planning started with the work of Drew McDermott and the UNPOP planner [20,22] (the acronym stands for *un-Partial Order Planner*). The planner proceeds forward in the state-space. Estimates of the distances between each state and the goals are based on the so-called *regression graph*, which is built from the goals using partially instantiated actions. UNPOP is similar to the GRT planner in its basic architecture, since it traverses the state-space in a forward direction and computes its estimates backwards. Its differences are that it does not consider the interactions between facts and that its heuristic is reconstructed for each state.

ASP [3] and HSP [4] (the acronyms stand for *Action Selection Planner* and *Heuristic Search Planner*, respectively) traverse the state-space and construct their heuristic in a forward direction. Their heuristic is based on the estimation of the distance between each fact of the problem and the current state, and is reconstructed for each state. The sum of the distance estimates of the goal facts is considered to be the cost of achieving them from the current state, thus usually resulting in overestimates.

A variation of HSP, named HSPr (*r* stands for regression) [5], constructs its heuristic once in a forward direction, whereas it traverses the state-space in a backward direction. The last member of the HSP family is HSP-2 [6], which supports a plethora of new heuristic functions, some of which being admissible [13].

The more recent planners are ALTALT [26] and FF [15]. They use planning graphs [2] to estimate the distances between states. The planning graphs are similar to the ASP/HSP/GRT heuristic, except for the fact that a max function is used instead of the sum or the AGGREGATE function, and mutual exclusion relations between facts are taken into account. ALTALT is a regression planner based on STAN [18] and HSPr. It creates a planning graph in a pre-processing phase and uses several techniques to extract heuristic estimates of the distances between the intermediate states and the initial state. For example, one heuristic returns the level in the planning graph, where all the facts of the current state appear, without any mutual exclusion relation between them.

FF traverses the state-space forward. In order to estimate the distance between an intermediate state and the goals, it creates a planning graph from each intermediate state to the goals, using *relaxed actions*, i.e., actions the delete lists of which have been removed. From this graph, FF extracts a *relaxed plan*, the length of which is the distance estimate.

7.2. Adaptation techniques

The techniques for multiobjective planning presented in this paper can be applied to all domain-independent heuristic planners. For the planners that compute individual distance estimates for the facts of a problem and then use a sum function, as in the case of UNPOP and the ASP/HSP family, the adaptation of the multiobjective paradigm is straightforward. Thus, these planners have to compute cost-vectors, instead of single values, and store all or some of the non-dominated ones for each fact (i.e., with or without the relaxed dominance pruning heuristic). Moreover, they have to continue computing vectors, until no fact can be achieved in a non-dominated way. However, in case a sum aggregate function is used, which results in significant over-estimates, the scales cannot be used for pruning

or penalizing the states of the state-space based on the heuristic estimates of the cost of the remaining plan.

For the planners that construct and exploit a planning graph, this graph must be transformed into a multiobjective one. In the original form of planning graphs, each fact-node was characterized by its level, which was a lower bound of the number of actions needed to be applied to the initial state, in order to achieve the fact. In the multiobjective planning graph, these levels play no important role any more. Now, a set of non-dominated cost-vectors, which correspond to the alternative paths achieving a fact, has to be assigned to each fact-node. The values of these vectors are lower bounds for the various criteria and for a specific path. Furthermore, for each cost-vector, the links to the cost-vectors of the preconditions of the action that achieved it, must be retained.

In terms of the mutual exclusion relations, two approaches can be adopted. The first one is to store these relations for each fact. The second and more powerful one is to store them for each alternative cost-vector of each fact. This means that a fact p may be mutually exclusive to another fact q, when p has been achieved in a way P_1 and q has been achieved in another way Q_1, while for alternative ways to achieve the two facts, these may be not mutually exclusive to each other. After the construction of the planning graph, the facts to which a specific fact p is mutually exclusive are produced by the intersection of the sub-sets of facts, to which p is mutually exclusive in its alternative cost-vectors.

The above approach can be adopted by non-heuristic graph-based planners, like STAN [18] and IPP [16]. As for the heuristic planners, which base the construction of their heuristics on planning graphs, the selection of the best cost-vectors of the current state facts (in the case of ALTALT) or the goal facts (in the case of FF) can be performed exactly in the same way as in the case of MO-GRT. For FF in particular, an approximate plan can be extracted by back-chaining from the best cost-vectors of the goal facts. However, this requires that the planning graph has been completely constructed, i.e., until no fact can be achieved in a non-dominated way.

8. Related work

The work presented in this paper is the first attempt to apply multiple-criteria evaluation techniques in the area of domain-independent heuristic planning. In the past, several works used multiple-criteria to evaluate plans. However, in those cases planning is either performed in a domain-dependent way, or demands exhaustive enumeration and evaluation of all states of the search-space.

For example, the PYRRHUS system [36], a partial order regression planner based on UCPOP [27], extends the definition of plan quality to consider partial satisfaction of the goal and the cost of resources used by the plan, while using domain specific heuristic knowledge for guidance. Other approaches, e.g., [10] and [24], are based on dynamic programming, which is inefficient for large search spaces, it guarantees however that the resulting plans will be optimal.

In some cases, utility models are also used, as in [11]. The extension of the framework presented in this paper, so as to cover probabilities and utility models, is straightforward.

However, there will be an overhead in the work of computing the estimates, since in that case the cost-vectors will have to be accompanied by their probabilities.

On the other hand, there are many approaches concerning planning with resources. However, here, as in the case of GRT-R [30] (an earlier version of MO-GRT), the goal is to find a solution plan not exceeding the available resources, whereas the criterion that is to be optimized is the length of the plan. For example, in [17], the IPP planning system [16] is extended in order to take resources into account. The new system, called RIPP, allows variable resource consumption and limited interdependencies between the resource variables of an action. The system annotates the fact nodes of a planning graph with resource intervals, which indicate the minimum and maximum available resource quantities, in order for a fact to be true in a specific time point.

Similarly, the LPSAT system [37] translates planning problems that employ resources in the LCNF formalism, where linear equalities and inequalities can be expressed, and uses the LPSAT solver, a systematic satisfiability solver integrated with an incremental SIMPLEX algorithm, to solve them. In this case too, the resources are only used as constraints, not as measures to be optimized.

Another approach is the EXCALIBUR system [25], where planning problems are formulated as structural constraint satisfaction problems (SCSPs). The system supports several types of constraints, such as *object constraints*, *action resource constraints*, *state resource constraints*, *task constraints*, *structural constraints* etc., which have to be hand-coded, and uses local search techniques along with global control to solve the problems. However, the inclusion of domain-specific knowledge to guide and accelerate the search for a plan is encouraged.

The TP4 system [14], which extends the work of Haslum and Geffner on admissible heuristics [13] in domains with actions that have duration, is closer to the framework proposed in this paper. The system guarantees the optimality of the resulting plans, in terms of the overall execution time (the *makespan*). However, resources are not combined in the heuristic function; instead, they are used separately for pruning purposes only. Furthermore, the authors recognize the problems that arise in domains where there is an interaction of time and resources and propose an integration of these measures, suggesting that the GRT-R approach is very promising.

9. Summary and future challenges

This paper presents MO-GRT, a heuristic state-space STRIPS planner, which extends the single objective planner GRT with the ability to take multiple criteria into account. The heuristic function is constructed in a domain-independent way, based only on the representation of the domain, i.e., the action schemata, and the definition of each problem, i.e., objects, initial state and goals.

MO-GRT takes a user-defined hierarchy of criteria, which are considered important for the resulting plan, some preferences among them, in the form of weights, and a scale of allowable values for each basic criterion. As the experimental results have shown, different weights and scales result in plans of different quality, with respect to the criteria, and in different planning times.

The work presented in this paper is the first attempt to apply multiple-criteria evaluation techniques in the area of domain-independent planning. The techniques presented for MO-GRT are general enough and can also be adopted by other modern heuristic state-space planners.

There are several challenges for future work in the area of multiobjective planning. The first one is the development of a meta-system, which will analyze a planning problem in an attempt to identify the boundaries where the cost of solving the problem with respect to the various criteria lies. The development of this system may require either some pre-processing planning (e.g., running the planner with marginal weights and collecting the solution plans) or domain-analysis techniques. This system would be useful to the evaluator, giving him the opportunity for a better understanding of the problem and helping him to set appropriate scales for the various criteria.

Another challenge refers to the limitations of the pure STRIPS representation. Modern domain-independent heuristic planners exhibit fine performance, however only in toy problems. Thus, planners should be enhanced, so as to handle more expressive languages, without losing their efficiency. We believe that MO-GRT is a step towards this direction.

Acknowledgements

The authors wish to thank the anonymous reviewers of the Artificial Intelligence Journal for their useful comments on the first version of this paper.

References

[1] F. Bacchus The official AIPS-00 planning competition home page, http://www.cs.toronto.edu/aips2000/.
[2] A.L. Blum, M.L. Furst, Fast planning through planning graph analysis, Artificial Intelligence 90 (1997) 281–300.
[3] B. Bonet, G. Loerincs, H. Geffner, A robust and fast action selection mechanism for planning, in: Proc. AAAI-97, Providence, RI, 1997, pp. 714–719.
[4] B. Bonet, H. Geffner, HSP: Heuristic search planner, in: 4th International Conf. on Artificial Intelligence Planning Systems (AIPS-98) Planning Competition, Pittsburgh, PA, 1998.
[5] B. Bonet, H. Geffner, Heuristic planning: New results, in: Proc. 5th European Conference on Planning (ECP-99), Durham, UK, 1999, pp. 359–371.
[6] B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence (Special Issue on Heuristic Search) 129 (1–2) (2001) 5–33.
[7] P. Dasgupta, P.P. Chakrabarti, S.C. DeSarkar, Multiobjective heuristic search in AND/OR graphs, J. Algorithms 20 (1996) 282–311.
[8] P. Dasgupta, P.P. Chakrabarti, S.C. DeSarkar, Searching game trees under a partial order, Artificial Intelligence 82 (1996) 237–257.
[9] R.E. Fikes, N.J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (1971) 189–208.
[10] K. Fujimura, Path planning with multiple objectives, IEEE Robotics and Automation Magazine 3 (1) (1996) 33–38.
[11] P. Haddawy, S. Hanks, Utility models for goal-directed decision-theoretic planners, Computational Intelligence 14 (3) (1998) 392–429.
[12] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Systems Sci. Cybernet. 4 (2), 100–107.

[13] P. Haslum, H. Geffner, Addmisible heuristics for optimal planning, in: Proc. 5th International Conference on AI Planning and Scheduling (AIPS-00), Breckenridge, CO, 2000, pp. 140–149.
[14] P. Haslum, H. Geffner, Heuristic planning with time and resources, in: Proc. 6th European Conference on Planning, Toledo, Spain, 2001.
[15] J. Hoffmann, B. Nebel, The FF planning system: Fast plan generation through heuristic search, J. Artificial Intelligence Res. 14 (2001) 253–302.
[16] J. Koehler, B. Nebel, J. Hoffmann, Y. Dimopoulos, Extending planning graphs to an ADL subset, in: Proceedings 4th European Conference on Planning (ECP-97), Durham, UK, 1997, pp. 273–285.
[17] J. Koehler, Planning under resource constraints, in: Proc. 13th European Conference on Artificial Intelligence (ECAI-98), Brighton, UK, 1998, pp. 489–493.
[18] D. Long, M. Fox, Efficient implementation of the plan graph in STAN, J. Artificial Intelligence Res. 10 (1998) 87–115.
[19] R.L. Keeney, H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, Wiley, New York, 1976.
[20] D. McDermott, A heuristic estimator for means-ends analysis in planning, in: Proc. 3rd Internat. Conf. on Artificial Intelligence Planning Systems (AIPS-96), Edinburgh, Scotland, 1996, pp. 142–149.
[21] D. McDermott, The official AIPS-98 planning competition home page, ftp://ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html.
[22] D. McDermott, Using regression-match graphs to control search in planning, Artificial Intelligence 109 (1–2) (1999) 111–159.
[23] M. Mollaghasemi, J. Pet-Edwards, Making Multiple-Objective Decisions, IEEE Computer Society, 1997.
[24] P. Moraitis, A. Tsoukias, Graph based representation of dynamic planning, in: Proc. 14th European Conference on Artificial Intelligence (ECAI-2000) Berlin, Germany, 2000, pp. 516–520.
[25] A. Nareyek, A, Beyond the plan-length criterion, in: A. Nareyek (Ed.), Local Search for Planning and Scheduling, in: Lecture Notes Artificial Intelligence, Vol. 2148, Springer, Berlin, 2001, pp. 55–78.
[26] R.S. Nigenda, X. Nguyen, S. Kambhampati, ALTALT: Combining the advantages of GRAPHPLAN and heuristic state search, in: Proc. International Conference on Knowledge-based Computer Systems, India, 2000.
[27] J. Penberthy, D. Weld, UCPOP: A sound and complete, partial order planner for ADL, in: Proc. KR-92, Cambridge, MA, 1992, pp. 103–114.
[28] I. Refanidis, I. Vlahavas, GRT: A domain independent heuristic for STRIPS worlds based on greedy regression tables, in: Proc. 5th European Conf. on Planning (ECP-99), Durham, UK, 1999, pp. 346–358.
[29] I. Refanidis, I. Vlahavas, L. Tsoukalas, On determining and completing incomplete states in STRIPS domains, in: Proc. IEEE International Conference on Information, Intelligence and Systems, Washington, DC, 1999, pp. 289–296.
[30] I. Refanidis, I. Vlahavas, Heuristic planning with resources, in: Proc. 14th European Conference on Artificial Intelligence (ECAI-00), Berlin, 2000, pp. 521–525.
[31] I. Refanidis, I. Vlahavas, The GRT planning system: Backward heuristic construction in forward state-space planning, J. Artificial Intelligence Res. 15 (2001) 115–161.
[32] I. Refanidis, I. Vlahavas, The GRT planner, AI Magazine 22 (3) (2001) 63–66.
[33] B.S. Stewart, C.C. White, Multiobjective A*, JACM 38 (4) (1991) 775–814.
[34] M. Veloso, Learning by analogical reasoning in general problem solving, Ph.D. Thesis, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1992.
[35] P. Vincke, Multi-Criteria Decision Aid, New York, 1992.
[36] M. Williamson, S. Hanks, Optimal planning with a goal-directed utility model, in: Proc. 2nd Int. Conf. on AI Planning Systems (AIPS-94), Chicago, IL, 1994, pp. 176–181.
[37] S.A. Wolfman, D. Weld, The LPSAT engine and its application to resource planning, in: Proc. IJCAI-99, Stockholm, Sweden, 1999, pp. 310–316.